2П

УДК 519.9

В.Н. Петрович
Киевский национальный университет имени Тараса Шевченко, г. Киев, Украина
filonval@i.com.ua

Разработка программного комплекса для оптимизации систем

В статье рассматривается программный комплекс для оптимизации систем. Комплекс использует объектно-ориентированный подход при реализации программ, имеет удобный интерфейс, не требует от пользователя знания особенностей языка программирования или деталей схемы организации процесса вычислений в комплексе.

Введение

Одной из особенностей применения задачи оптимизации к задаче идентификации параметров динамической модели, представляющей собой систему дифференциальных уравнений, по критерию минимизации несогласованности переходных является тот факт, что в таком случае вычисления значения $f(x)$ требуют значительных вычислительных затрат, что затрудняет нахождение на *k*-й итерации $f(x_k)$ и делает почти невозможным вычисления значения производных $f'(x_k)$. Это накладывает определенные ограничения на класс допустимых алгоритмов оптимизации при решении задачи идентификации параметров модели. Данное замечание касается и реальных задач оптимизации технологических режимов функционирования объектов, где в большинстве случаев трудно в явном аналитическом виде представить критерий оптимизации.

Эти соображения стали основой при выборе множества алгоритмов оптимизации, которые представлены в разработанном комплексе программ. Каждый из существующих алгоритмов многомерной оптимизации имеет свои преимущества и недостатки и может более эффективно работать или в окрестности точки минимума, или на начальных этапах оптимизации, когда приближение находится далеко от точки минимума.

Целью данной работы является разработка комплекса алгоритмов оптимизации для последовательного подключения различных алгоритмов на тех этапах процесса оптимизации, где каждый из них будет наиболее эффективным. Существенной помощью в выборе эффективного метода в случае конкретной функции $f(x)$ может стать предусмотренная в комплексе программ возможность графического представления линий уровня исследуемой функции $f(x)$.

Постановка задачи минимизации

Разработанная подсистема позволяет определять точку минимума заданной функции многих переменных $f(x), x=(x_1,\dots,x_n)$ для задачи без ограничений

$$\arg\min_{x\in E_n} f(x),$$

или для задачи с ограничениями

$$\arg\min_{x\in D} f(x),\quad D=\left\{x: a_i \le x_i \le b_i, \overline{i=1,n}\right\}.$$

Ограничения на параметры диктуются самой постановкой задачи, исходя из физических соображений.

Общая схема реализации алгоритмов минимизации

В вычислительных схемах большинства алгоритмов минимизации можно выделить ряд общих последовательных этапов, а именно:

1) выбор начального приближения x_0 (нулевая итерация $k = 0$) и вычисления $f(x_0)$;

2) определение направления спуска p_k для минимизации $f(x)$ (способ решения этой подзадачи определяющий, для каждого конкретного алгоритма характерна своя стратегия при выборе направления спуска);

3) определение точки минимума $x_{\min}^{(k)}$ на луче:

$$x = x_k + \alpha \cdot h_{k\cdot} p_k : \min_{\alpha>0} f(x_k + \alpha \cdot h_{k\cdot} p_k) = f(x_k + \alpha_{\min} h_k p_k);$$

4) переопределение полученной точки минимума на луче $x_{k+1} = x_k + \alpha_{\min} \cdot h_{k\cdot} p_k$ в качестве исходной для следующей итерации $k = k+1$;

5) условный переход на пункт 2, если критерий завершения (выхода из алгоритма) не выполняется. Однократное выполнение этапов 2 – 5 будем называть одной итерацией процесса минимизации. На каждом из этапов нужно фактически решить некоторую самостоятельную подзадачу. Для решения подзадач 1 и 3 в комплексе предусмотрено несколько альтернативных методов, которые реализованы в виде самостоятельных программных модулей.

Выбор начальной точки x_0 осуществляется по такой схеме: если начальная точка x_0 пользователем не задана, то автоматически выбирается и случайная точка из множества $l = n+1$ равномерно распределенных точек ξ_k, $k = \overline{1,l}$ в некоторой заданной области D_0 – n-мерном параллелепипеде: $D_0 = \left\{x : a_i^0 \le x_i \le b_i^0, \overline{i = 1,n}\right\}$, где достигается $\min_{1\le k\le l} f(\xi_k)$.

На этапе 5 в понятие «критерий выхода» укладывается проверка выполнения двух условий:

1) алгоритм «исчерпал себя», что отразилось на выполнении так называемого «внутреннего» критерия (этот критерий свой для каждого алгоритма), тогда выход из программы минимизации происходит с кодом возврата $kw = 1$;

2) были выполнены заданное число *KIT* итераций, максимум на которые был запущен данный метод (точнее модуль его реализует), однако внутренний критерий не исполнился, тогда возвращение из модуля с кодом $kw = 2$. Число итераций *KIT* задается пользователем или по умолчанию выбирается управляющей программой. При возвращении в управляющую программу минимизации с $kw = 2$ осуществляется анализ скорости сходимости алгоритма минимизации. Для этого используется собранная в процессе выполнения последних *MIT* итераций информация о значении $\{x_k, f(x_k)\}$, $k = \overline{1, MIT}$. Эта информация собирается и накапливается в глобальной структуре *STAT*, которая подключена к программным модулям минимизации. Потом идет проверка условия $R_X > VA \cup R_Y > VK$, где

$$R_X = \frac{1}{MIT}\sqrt{\sum_{k=1}^{MIT-1}\left(\sum_{i=1}^{m}(x_k^{(i)} - x_{k+1}^{(i)})^2\right)},$$

$$R_Y = \frac{1}{MIT}\sqrt{\sum_{k=1}^{MIT-1}(f(x_k) - f(x_{k+1}))^2}.$$

Если «нет», то скорость сходимости данного алгоритма неудовлетворительная и управляющая программа на следующие *KIT* итераций автоматически подключит следующий алгоритм минимизации, который предусмотрен в цепочке алгоритмов,

которыми планируется решать данную задачу минимизации. Эта цепочка или «по умолчанию» назначается автоматически управляющей программой, или может быть задана перед началом процесса решения задачи пользователем. Благодаря этому в случае неудовлетворительной сходимости пользователь может управлять алгоритмами, изменяя значения их входных параметров или конфигурацию модулей, включенных в цепочку, и описывать конфигурацию алгоритма.

Если это условие выполняется, то алгоритм с данной конфигурацией модулей и значениями их входных параметров запускается повторно без изменений по следующим *KIT* итерациям. Этот процесс выбора алгоритма на следующее число итераций осуществляется до тех пор, пока с алгоритма не произойдет выход по его внутреннему критерию или заданная цепочка алгоритмов не исчерпает своих возможностей по успешной минимизации функции.

Значения *MIT*, *VA*, *VK* и *KIT* задаются пользователем или автоматически определяются «по умолчанию» управляющей программой. Конфигурация алгоритма решения задачи, задания значений входных параметров составляющих модулей минимизации, а также описание цепочки решающих алгоритмов дается в специальной структуре *TALG*.

Множество алгоритмов минимизации

Для нахождения решения вышепоставленных задач минимизации предусмотрена возможность использования любого допустимого для данной задачи метода, реализованного в комплексе программ набора методов по нахождению минимума многомерной и одномерной функции. Пользователь имеет возможность по своему усмотрению выбирать любой из следующих методов многомерной минимизации:

1 МСЛН – метод случайных направлений;
2 МГРД – метод градиентного спуска;
3 МСМП – симплекс метод;
4 МСГР – метод сопряженных градиентов;
5 МПАУ – метод Пауэла;
6 МСЛМ – модифицированный метод случайных направлений;
7 МОВР – градиентный метод овражного типа;
8 МГРС – градиентный метод с растяжением пространства;
9 МПКС – метод покоординатного спуска;
10 МФПД – метод Флетчера – Пауэла – Давидона;
11 МГРО – метод градиентного спуска с ограничениями;
12 МСЛО – метод случайных направлений с ограничениями.

Для реализации одномерного поиска точки минимума в заданном направлении комбинировать его с одним из методов одномерной минимизации:

1 ПАРРТ – метод парабол по равномерным точкам;
2 ФПАРБ – метод парабол с золотым сечением;
3 ФИБОН – метод Фибоначчи.

При решении задачи минимизации целесообразно комбинировать комплексное использование алгоритмов на разных этапах алгоритма минимизации в зависимости от свойств конкретного алгоритма. С этой точки зрения представленные в подсистеме алгоритмы можно классифицировать следующим образом.

На начальном этапе поиска, когда точка начального приближения x_0 находится далеко от точки минимума $x_{\min}$, целесообразно применять методы линейной аппроксимации, которые требуют минимальных вычислительных затрат, как, например, градиентный метод, метод случайных направлений, модифицированный метод случайных направлений. Все эти методы имеют линейную скорость сходимости и за сравнительно небольшое количество итераций позволяют приблизиться в окрестность точки минимума или спуститься на дно «оврага», где скорость сходимости этих методов замедляется.

При структуре линий уровня $f(x) = \text{const}$ типа «желоба» (вогнутость линий будет в направлении малочувствительных компонент) сходимость всех перечисленных методов будет плохой. В таком случае возможно «исправление» поверхности путем изменения масштабов переменных при применении градиентного метода с растяжением пространства или применения градиентного метода «овражного» типа.

При приближении к точке минимума с помощью линейных методов происходит замедление сходимости. И поэтому здесь целесообразно переходить на методы минимизации второго порядка, использующие квадратичную аппроксимацию $f(x)$. Из числа методов такого типа в комплексе представлены: метод Пауэла и метод Флетчера – Пауэла – Давидона. Эти методы минимизируют квадратичную функцию; на строго выпуклых дважды непрерывных дифференцируемых функциях имеют квадратичную сходимость; без предположения строгой выпуклости методы имеют надлинейную сходимость. Можно также использовать метод сопряженных градиентов, который хоть и является методом первого порядка, но имеет более высокую сходимость по сравнению с методами градиентного типа и для квадратичной функции, как и методы Флетчера и Флетчера – Пауэла – Давидона, совпадает не более чем по n итераций.

Каждый из этих методов реализован в виде отдельного модуля, который может быть использован как стандартная подпрограмма из библиотеки модулей и может быть подключен к задаче минимизации через посредничество предусмотренного в комплексе интерфейса пользователя, что позволяет при решении конкретной задачи свести к минимуму этап программирования.

Организация модулей для задачи оптимизации

После старта управляющей программы *strtmn()* пользователю предоставляется возможность с помощью диалоговых процедур задать функцию, которую он хочет минимизировать и определить (при желании) алгоритм минимизации. Модуль *strtmn()* осуществляет подготовку к началу процесса решения задачи и инициализирует структуры: *tzad* – с информацией о поставленной задаче; *tmod* – с описанием модели, в рамках которой решается задача; *talg* – с описанием алгоритма решения задачи и значениями входных параметров модулей минимизации; *prnt* – с описанием режимов документирования задачи. Описание этих структур можно найти в файле *tblcls.cpp*. Далее в модуле *strtmn()* инициализируется заданная функция минимизации *init_min()* и происходит задание значений параметров «по умолчанию» и конфигурации алгоритма решения задачи.

За вызов необходимого метода многомерной оптимизации отвечает модуль *mnmin.cpp*, методы одномерной минимизации вызываются модулем *momin.cpp*, модуль *lomin.cpp* отвечает за локализацию точки минимума в заданном направлении. Модуль *momin.cpp* вызывается модулем *mnmin.cpp* в том случае, если выход из подпрограммы минимизации произошел не из-за нахождения точки минимума с заданной точностью, а через заданное число итераций, т.е. если имеется плохая сходимость алгоритма. Об этом случае свидетельствует значение параметра $kw = 2$.

Описание процесса разработки модулей своей задачи минимизации

Он включает следующие этапы:

Создать модуль целевой функции (на основе шаблона функции *f()*): – вместо *f* дать имя модуля целевой функции $y = f(x)$.

В операторе $y = ...$ записать формулу для вычисления $f(x)$. Создать модуль описания и инициализации задачи (на основе шаблона функции *initf ()*):

– в $n = ...$ указать размерность вектора x, а в описании *char * name_x[]* – наименование компонент (для документирования результатов);

– если задана начальная точка $x0$, то в описании *double x0[] = ()* задается ее числовое значение и флажок $t.ix0 = 1$, иначе под $x0$ только резервируется память *double x0[n]* и $t.ix0 = 0$;

– если заданная область начальной точки x0, то включаем описание массивов *double da0[]* = (), *double db0[]* = (), где задаем числовые значения области поиска начальной точки: $da0 < x0 < db0$, и значение флажком $t.id0 = 1$;

– если заданная область ограничений на вектор x, то включаем описание массивов *double da[] = ()*, *double db[] = ()*, в которых задаем числовые значения области поиска точки минимум: $da < x < db$, и значение флажком $t.id = 1$.

```
/ / «МОДУЛЬ ЦЕЛЕВОЙ ФУНКЦИИ F (X) = ...»,
double f (double * x)(double y; y = ... ;
return y;
)
/ * МОДУЛЬ инициализации ЗАДАЧИ * /
TMIN (
int n; int ix0 = 0; int id0 = 0; int id = 0;
double * x0;
double * da0, db0; double * da, db;
char * name [];
)
void initf (void)
(
void initmin (TMIN &);
TMIN t;
static double x0 [] = (); static double da0 [] = ();
static double db0 [] = ();static double da [] = ();
static double db [] = ();
static char * name_x []={« X [1] «,» X [2]);
t.n =; t.ix0 =;t.id0 =; t.id =;
t-> name = name_x;
initmin (& t);
return;
)
```

Заключение

Разработанный комплекс предназначен для оптимизации систем и благодаря использованию объектно-ориентированного подхода при реализации программ имеет удобный интерфейс, не требует от пользователя знания особенностей языка программирования или деталей схемы организации процесса вычислений в комплексе.

Литература

1. Сучасні методи і інформаційні технології математичного моделювання, аналізу й оптимізації складних систем / [Ф.Г. Гаращенко, М.Ф. Кириченко, Ю.В. Крак та ін.]. – К. : ВПЦ «Київський університет», 2006. – 200 с.
2. Бард Й. Нелинейное оценивание параметров / Бард Й. – М. : Статистика, 1979. – 349 с.
3. Гилл Ф. Практическая оптимизация / Гилл Ф., Мюррей У., Райт М. – М. : Мир, 1977. – 290 с.
4. Деннис Дж. Численные методы безусловной оптимизации и решения нелинейных уравнений / Дж. Деннис, Р. Шнабель. – М. : Мир, 1988. – 440 с.
5. Моисеев Н.Н. Численные методы в теории оптимальных систем / Моисеев Н.Н. – М., 1971. – 424 с.

В.М. Петрович

Розробка програмного комплексу для оптимізації систем

У статті розглядається програмний комплекс для оптимізації систем. Комплекс використовує об'єктно-орієнтований підхід при реалізації програм, має зручний інтерфейс, не вимагає від користувача знання особливостей мови програмування або деталей схеми організації процесу обчислень в комплексі.

V.N. Petrovich

Development of Software for Optimization System

The article discusses a software package for optimizing systems. The complex uses the object-oriented approach to programming, has a convenient interface that does not require any knowledge of the programming language or the details of the schemes of computation in the complex.

Статья поступила в редакцию 08.07.2010.